(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 116450737 B
(45)授权公告日 2025.06.06

(21)申请号 202310340844.X
(22)申请日 2023.03.31
(65)同一申请的已公布的文献号
申请公布号 CN 116450737 A
(43)申请公布日 2023.07.18
(73)专利权人 山东大学
地址 266237 山东省青岛市即墨滨海路72号
(72)发明人 李梁 栾昊
(74)专利代理机构 济南圣达知识产权代理有限公司 37221
专利代理师 黄海丽
(51)Int.Cl.
G06F 16/27(2019.01)
G06F 18/25(2023.01)
G06F 21/60(2013.01)
G06F 16/23(2019.01)
(56)对比文件
CN 112424810 A,2021.02.26
CN 111832034 A,2020.10.27
审查员 高悦

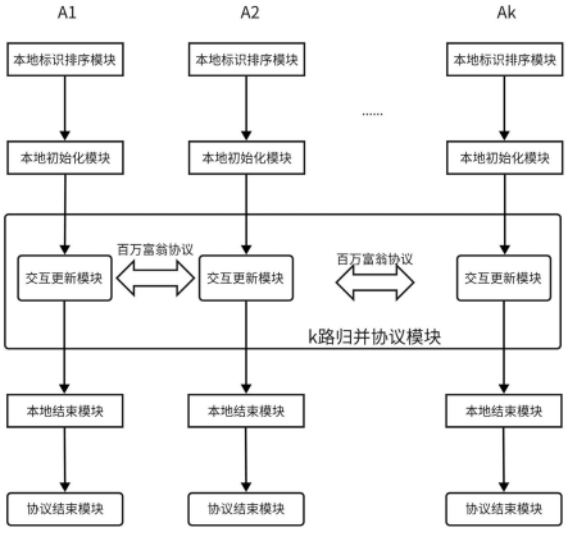
权利要求书2页 说明书8页 附图2页

(54)发明名称

基于堆结构的安全多方数据同步预处理方法及系统

(57)摘要

本公开提供了一种基于堆结构的安全多方数据同步预处理方法及系统,其应用于若干数据端之间的多方数据融合,包括:对于每个数据端,按照其本地数据标识大小对数据进行本地排序,获得本地数据列表;对于每个数据端,分别定义指向本地数据列表第一个元素标识的位置指针,并构建最终数据同步列表,其中,所述最终数据同步列表初始为空列表;基于各个数据端的位置指针所指向的标识构建最小堆;通过遍历各个数据端中的本地数据列表,循环更新最小堆以及最终数据同步列表,直至各个数据端中的本地数据列表均遍历至结尾,结束循环;以获得的每个终端的最终数据同步列表作为多方数据的融合结果。



CN 116450737 B

1.一种基于堆结构的安全多方数据同步预处理方法,其特征在于,其应用于若干数据端之间的多方数据融合,包括:

对于每个数据端,按照其本地数据标识大小对数据进行本地排序,获得本地数据列表;

对于每个数据端,分别定义指向本地数据列表第一个元素标识的位置指针,并构建最终数据同步列表,其中,所述最终数据同步列表初始为空列表;

基于各个数据端的位置指针所指向的标识构建最小堆;

通过遍历各个数据端中的本地数据列表,循环更新最小堆以及最终数据同步列表,直至各个数据端中的本地数据列表均遍历至结尾,结束循环;

以获得的每个终端的最终数据同步列表作为多方数据的融合结果;

其中,所述循环更新最小堆以及最终数据同步列表,具体为:获取最小堆结构中的最小元,将最小元标识对应的数据添加至该标识对应数据端的最终数据同步列表的末端;更新最小元标识对应数据端的位置指针指向下一位置;在最小堆中删除当前最小元,并增加当前最小元标识对应数据端位置指针所指向的标识,实现最小堆的更新;若更新后最小堆的最小元标识与前一个删除的最小元相等,则重新执行下一轮循环,若不等,则在本轮循环中未添加数据的数据端最终数据同步列表的末端添加随机数据。

2.如权利要求1所述的一种基于堆结构的安全多方数据同步预处理方法,其特征在于,所述最小堆的初始化过程中,每个数据端不需要存储完整的最小堆,仅需存储自身当前位置指针所指向标识在最小堆中的父节点和子节点;

或,

所述最小堆的初始化过程中,每个数据端各自维护完整的最小堆。

3.如权利要求1所述的一种基于堆结构的安全多方数据同步预处理方法,其特征在于,所述在最小堆中删除当前最小元,并增加当前最小元标识对应数据端位置指针所指向的标识,具体为:将最小堆中最小元结点对应的标识直接更新为新增加的标识,并更新此堆使其成为最小堆。

4.如权利要求3所述的一种基于堆结构的安全多方数据同步预处理方法,其特征在于,所述更新此堆使其成为最小堆,具体为:

步骤1:若更新的结点有父结点,更新的结点和其父结点对应的数据端交互比较大小;若更新的结点更大,进入步骤2;若更新的结点更小,交换此结点与父结点,并通知父结点对应的数据端更新自己的位置,返回步骤1继续与父结点比较;

步骤2:若更新的结点有子结点,分别与左子结点和右子结点对应的数据端交互比较大小;若更新的结点更大,交换此结点与该子结点,并通知该子结点对应的数据端更新自己的位置,返回步骤2继续与子结点比较;若更新的结点更小,进入步骤3;

步骤3:通知所有数据端自己更新后的位置。

5.如权利要求1所述的一种基于堆结构的安全多方数据同步预处理方法,其特征在于,在最小堆的构建及更新过程中,对于每个数据端,实时存储其当前位置指针所指向的标识,以及该标识在最小堆中的父节点和子节点对应的数据端编号。

6.如权利要求1所述的一种基于堆结构的安全多方数据同步预处理方法,其特征在于,在最小堆更新时,基于各个数据端中存储的父节点和子节点编号,寻找相应的数据端;不同数据端之间通过编号查找进行两两标识的比较,其中,所述两两标识的比较采用基于百万

富翁难题的多方安全计算。

7.如权利要求6所述的一种基于堆结构的安全多方数据同步预处理方法,其特征在于,所述两两标识的比较采用基于百万富翁难题的多方安全计算,具体为:

对于两个待比较标识分别对应的第一数据端和第二数据端,第一数据端通过百万富翁协议的加密自己的标识,记为第一加密标识,发送给第二数据端;

第二数据端将自己的标识和收到的第一加密标识基于百万富翁协议进行比较大小,获得比较结果;将自身标识通过百万富翁协议加密得到第二加密标识,并将其发送给第一数据端;

第一数据端将自己的标识和收到的第二加密标识基于百万富翁协议进行比较大小,获得比较结果。

8.一种基于堆结构的安全多方数据同步预处理系统,其特征在于,其应用于若干数据端之间的多方数据融合,包括:

本地排序单元,其用于对于每个数据端,按照其本地数据标识大小对数据进行本地排序,获得本地数据列表;

最终数据同步列表初始化单元,其用于对于每个数据端,分别定义指向本地数据列表第一个元素标识的位置指针,并构建最终数据同步列表,其中,所述最终数据同步列表初始为空列表;

最小堆构建单元,其用于基于各个数据端的位置指针所指向的标识构建最小堆;

多方数据融合单元,其用于通过遍历各个数据端中的本地数据列表,循环更新最小堆以及最终数据同步列表,直至各个数据端中的本地数据列表均遍历至结尾,结束循环;以获得的每个终端的最终数据同步列表作为多方数据的融合结果;

其中,所述循环更新最小堆以及最终数据同步列表,具体为:获取最小堆结构中的最小元,将最小元标识对应的数据添加至该标识对应数据端的最终数据同步列表的末端;更新最小元标识对应数据端的位置指针指向下一位置;在最小堆中删除当前最小元,并增加当前最小元标识对应数据端位置指针所指向的标识,实现最小堆的更新;若更新后最小堆的最小元标识与前一个删除的最小元相等,则重新执行下一轮循环,若不等,则在本轮循环中未添加数据的数据端最终数据同步列表的末端添加随机数据。

9.一种电子设备,包括存储器、处理器及存储在存储器上运行的计算机程序,其特征在于,所述处理器执行所述程序时实现如权利要求1-7任一项所述的一种基于堆结构的安全多方数据同步预处理方法。

10.一种非暂态计算机可读存储介质,其上存储有计算机程序,其特征在于,该程序被处理器执行时实现如权利要去1-7任一项所述的一种基于堆结构的安全多方数据同步预处理方法。

基于堆结构的安全多方数据同步预处理方法及系统

技术领域

[0001] 本公开属于计算机技术领域,尤其涉及一种基于堆结构的安全多方数据同步预处理方法及系统。

背景技术

[0002] 本部分的陈述仅仅是提供了与本公开相关的背景技术信息,不必然构成在先技术。

[0003] 多方数据融合是目前许多商业公司、企业和机构广泛关注的技术问题。它是通过多方安全计算协议来实现数据同步,让不同的数据持有者都有各方的所有数据样本,但不暴露任何一方数据的具体信息。例如:多家医院联合使用各自的病例信息进行更准确的诊断;多家金融机构联合使用各自的信用记录来发现潜在的金融风险等等。因为在多方数据融合中,所有数据持有者都不想暴露自己的数据隐私,即用户级隐私需要得到严格保障。并且,多方数据融合可以提高未来基于数据分析的模型质量或问题挖掘深度。

[0004] 发明人发现,现有得多方数据融合方法存在以下缺点:

[0005] 大多数现有技术都是使用诸如遗忘传输(oblivious transfer)之类的复杂技术实现的,而这中策略算法实现复杂,在实际中运行效率较低;其次,现有技术对于仅找到双方的交集结果虽然是可行的,但实际中多方数据融合不仅仅只有两方,更多的情况是多于两方企业进行数据融合,且希望找到多方数据的并集;同时,当现有技术用于多方融合时,方案非常复杂,需要多次调用两方数据融合的算法,成本较高,不利于实际应用。

发明内容

[0006] 本公开为了解决上述问题,提供了一种基于堆结构的安全多方数据同步预处理方法及系统,所述方案通过使用基于百万富翁难题的安全多方计算比较大小的协议以及数据归并策略,大大降低了多方数据融合的复杂性,同时,所述方案能够达到数据融合的最终效果,使得数据按照标识大小排列,并严格满足数据的安全要求。

[0007] 根据本公开实施例的第一个方面,提供了一种基于堆结构的安全多方数据同步预处理方法,其应用于若干数据端之间的多方数据融合,包括:

[0008] 对于每个数据端,按照其本地数据标识大小对数据进行本地排序,获得本地数据列表;

[0009] 对于每个数据端,分别定义指向本地数据列表第一个元素标识的位置指针,并构建最终数据同步列表,其中,所述最终数据同步列表初始为空列表;

[0010] 基于各个数据端的位置指针所指向的标识构建最小堆;

[0011] 通过遍历各个数据端中的本地数据列表,循环更新最小堆以及最终数据同步列表,直至各个数据端中的本地数据列表均遍历至结尾,结束循环;

[0012] 以获得的每个终端的最终数据同步列表作为多方数据的融合结果;

[0013] 其中,所述循环更新最小堆以及最终数据同步列表,具体为:获取最小堆结构中的

最小元,将最小元标识对应的数据添加至该标识对应数据端的最终数据同步列表的末端;更新最小元标识对应数据端的位置指针指向下一位置;在最小堆中删除当前最小元,并增加当前最小元标识对应数据端位置指针所指向的标识,实现最小堆的更新;若更新后最小堆的最小元标识与前一个删除的最小元相等,则重新执行下一轮循环,若不等,则在本轮循环中未添加数据的数据端最终数据同步列表的末端添加随机数据。

[0014] 进一步的,所述最小堆的初始化过程中,每个数据端不需要存储完整的最小堆,仅需存储自身当前位置指针所指向标识在最小堆中的父节点和子节点;

[0015] 或,

[0016] 所述最小堆的初始化过程中,每个数据端各自维护完整的最小堆。

[0017] 进一步的,所述在最小堆中删除当前最小元,并增加当前最小元标识对应数据端位置指针所指向的标识,具体为:将最小堆中最小元结点对应的标识直接更新为新增加的标识,并更新此堆使其成为最小堆。

[0018] 进一步的,所述更新此堆使其成为最小堆,具体为:

[0019] 步骤1:若更新的结点有父结点,更新的结点和其父结点对应的数据端交互比较大小;若更新的结点更大,进入步骤2;若更新的结点更小,交换此结点与父结点,并通知父结点对应的数据端更新自己的位置,返回步骤1继续与父结点比较;

[0020] 步骤2:若更新的结点有子结点,分别与左子结点和右子结点对应的数据端交互比较大小;若更新的结点更大,交换此结点与该子结点,并通知该子结点对应的数据端更新自己的位置,返回步骤2继续与子结点比较;若更新的结点更小,进入步骤3。

[0021] 步骤3:通知所有数据端自己更新后的位置。

[0022] 进一步的,在最小堆的构建及更新过程中,对于每个数据端,实时存储其当前位置指针所指向的标识,以及该标识在最小堆中的父节点和子节点对应的数据端编号。

[0023] 进一步的,在最小堆更新时,基于各个数据端中存储的父节点和子节点编号,寻找相应的数据端;不同数据端之间通过编号查找进行两两标识的比较,其中,所述两两标识的比较采用基于百万富翁难题的多方安全计算。

[0024] 进一步的,所述两两标识的比较采用基于百万富翁难题的多方安全计算,具体为:

[0025] 对于两个待比较标识分别对应的第一数据端和第二数据端,第一数据端通过百万富翁协议的加密自己的标识,记为第一加密标识,发送给第二数据端;

[0026] 第二数据端将自己的标识和收到的第一加密标识基于百万富翁协议进行比较大小,获得比较结果;将自身标识通过百万富翁协议加密得到第二加密标识,并将其发送给第一数据端;

[0027] 第一数据端将自己的标识和收到的第二加密标识基于百万富翁协议进行比较大小,获得比较结果。

[0028] 根据本公开实施例的第二个方面,提供了一种基于堆结构的安全多方数据同步预处理系统,其应用于若干数据端之间的多方数据融合,包括:

[0029] 本地排序单元,其用于对于每个数据端,按照其本地数据标识大小对数据进行本地排序,获得本地数据列表;

[0030] 最终数据同步列表初始化单元,其用于对于每个数据端,分别定义指向本地数据列表第一个元素标识的位置指针,并构建最终数据同步列表,其中,所述最终数据同步列表

初始为空列表；

[0031] 最小堆构建单元，其用于基于各个数据端的位置指针所指向的标识构建最小堆；

[0032] 多方数据融合单元，其用于通过遍历各个数据端中的本地数据列表，循环更新最小堆以及最终数据同步列表，直至各个数据端中的本地数据列表均遍历至结尾，结束循环；以获得的每个终端的最终数据同步列表作为多方数据的融合结果；

[0033] 其中，所述循环更新最小堆以及最终数据同步列表，具体为：获取最小堆结构中的最小元，将最小元标识对应的数据添加至该标识对应数据端的最终数据同步列表的末端；更新最小元标识对应数据端的位置指针指向下一位置；在最小堆中删除当前最小元，并增加当前最小元标识对应数据端位置指针所指向的标识，实现最小堆的更新；若更新后最小堆的最小元标识与前一个删除的最小元相等，则重新执行下一轮循环，若不等，则在本轮循环中未添加数据的数据端最终数据同步列表的末端添加随机数据。

[0034] 根据本公开实施例的第三个方面，提供了一种电子设备，包括存储器、处理器及存储在存储器上运行的计算机程序，所述处理器执行所述程序时实现所述的一种基于堆结构的安全多方数据同步预处理方法。

[0035] 根据本公开实施例的第四个方面，提供了一种非暂态计算机可读存储介质，其上存储有计算机程序，该程序被处理器执行时实现所述的一种基于堆结构的安全多方数据同步预处理方法。

[0036] 与现有技术相比，本公开的有益效果是：

[0037] (1)本公开提供了一种基于堆结构的安全多方数据同步预处理方法及系统，所述方案通过使用基于百万富翁难题的安全多方计算比较大小的协议以及数据归并策略，大大降低了多方数据融合的复杂性，同时，所述方案能够达到数据融合的最终效果，使得数据按照标识大小排列，并严格满足数据的安全要求。

[0038] (2)本公开所述方案可以应用在多方数据融合的情况，不再是两两数据融合再合并的方法，并且复杂性和成本不高，便于推广应用。

[0039] 本公开附加方面的优点将在下面的描述中部分给出，部分将从下面的描述中变得明显，或通过本公开的实践了解到。

附图说明

[0040] 构成本公开的一部分的说明书附图用来提供对本公开的进一步理解，本公开的示意性实施例及其说明用于解释本公开，并不构成对本公开的不当限定。

[0041] 图1为本公开实施例中所述的多方数据融合案例示意图；

[0042] 图2为本公开实施例中所述的最小堆的初始化过程示意图；

[0043] 图3为本公开实施例中所述的基于堆结构的安全多方数据同步预处理方法的整体流程图。

具体实施方式

[0044] 下面结合附图与实施例对本公开做进一步说明。

[0045] 应该指出，以下详细说明都是例示性的，旨在对本公开提供进一步的说明。除非另有指明，本文使用的所有技术和科学术语具有与本公开所属技术领域的普通技术人员通常

理解的相同含义。

[0046] 需要注意的是,这里所使用的术语仅是为了描述具体实施方式,而非意图限制根据本公开的示例性实施方式。如在这里所使用的,除非上下文另外明确指出,否则单数形式也意图包括复数形式,此外,还应当理解的是,当在本说明书中使用术语“包含”和/或“包括”时,其指明存在特征、步骤、操作、器件、组件和/或它们的组合。

[0047] 在不冲突的情况下,本公开中的实施例及实施例中的特征可以相互组合。

[0048] 术语解释:

[0049] 多方安全计算:指在无可信第三方的情况下,多方各自持有一部分数据,通过安全通信协议交互计算一个约定函数。整个通信协议需要满足以下条件:

[0050] 1.在安全通信协议结束后,参与协议的所有各方都可以获得协议中计算得到的函数值;

[0051] 2.在安全通信协议的整个过程中以及结束后,协议中涉及的所有各方都无法获得关于任何其他方数据的任何信息。

[0052] 最小堆:最小堆,是一种经过排序的完全二叉树,其中任一结点的数据值均不大于它左子结点和右子结点的数值。本公开所述方案最小堆中记录的不是真实数据值,而是数据端的编号,大小的比较使用的是数据端各自的标识比较,而比较过程使用的是基于百万富翁协议的安全计算来比较大小。

[0053] 百万富翁难题:百万富翁难题由华人世界唯一图灵奖得主姚期智先生提出,是多方安全计算领域的源头。该问题具体描述为:假设A和B分别持有数据Xa和Xb,需要在没有可信第三方的情况下,计算Xa和Xb那个数值更大,且比较完毕后A和B都得到哪一个更大的信息而没获得另一方数据的任何信息。目前已经有很多可实现的安全计算方案(RSA,混淆电路等)解决上述百万富翁难题。本公开所述方案使用该问题的解决方案来完成多方安全比较大小。

[0054] 多方数据同步:是指在多个参与方拥有不同的数据样本或同一数据样本的不同特征的情况下,通过使用多方安全计算协议来实现数据的同步。数据同步的最终结果将包括所有各方的数据样本和所有各方数据的特征维度。换言之,它旨在实现跨多个数据源的数据集成和分析。本发明中使用的是没有可信第三方情况下的多方数据同步。

[0055] 数据对齐:在多个数据持有者之间,每个数据样本点都有唯一的序号或标识,可以将各自的数据按照标识排列,以便在数据融合之前进行数据对齐。在进行数据融合之后,各方可以获得所有数据,但仅知道自己方序号或标识,而不会暴露其他任何方的序号或标识。

[0056] 实施例一:

[0057] 本实施例的目的是提供一种基于堆结构的安全多方数据同步预处理方法。

[0058] 为了便于理解,首先对本实施例所述方案所要解决的技术问题进行举例说明:

[0059] 如图1所示,以下通过举例说明多方(以三方为例)数据融合需要完成的任务。其中A、B和C分别代表不同的三方数据端,key_1到key_9是各端持有数据的标识(key),dataXX是各端不同标识对应的数据,“*”为各端自己无法识别的标识数据。下图例子中,数据同步融合之前,A持有标识key_1、key_2、key_6、key_7和key_9,以及相对应的数据;B持有标识key_2、key_3、key_5、key_8和key_9,以及相对应的数据;C持有标识key_2、key_4、key_7、key_8和key_9,以及相对应的数据。最终,通过多方数据同步融合的协议,A端持有所有的9个数

据,但仅知道自己原有数据所对应的key,其余非原有数据的标识为任意乱码或随机串。同理其他所有方也持有所有的9个数据,仅知道自己持有数据对应的key,自己没有数据的无法获得任何信息。注意的是,三方数据的并集为9个,也就是最终三方都持有数据并集,但并不知道对方数据的标识。后续每一方要使用其他方数据的时候,可以通过遗忘传输协议与数据持有端发起通信,进而在保证隐私的前提下获得数据。

[0060] 一种基于堆结构的安全多方数据同步预处理方法,其应用于若干数据端之间的多方数据融合,包括:

[0061] 对于每个数据端,按照其本地数据标识大小对数据进行本地排序,获得本地数据列表;

[0062] 对于每个数据端,分别定义指向本地数据列表第一个元素标识的位置指针,并构建最终数据同步列表,其中,所述最终数据同步列表初始为空列表;

[0063] 基于各个数据端的位置指针所指向的标识构建最小堆;

[0064] 通过遍历各个数据端中的本地数据列表,循环更新最小堆以及最终数据同步列表,直至各个数据端中的本地数据列表均遍历至结尾,结束循环;

[0065] 以获得的每个终端的最终数据同步列表作为多方数据的融合结果;

[0066] 其中,所述循环更新最小堆以及最终数据同步列表,具体为:获取最小堆结构中的最小元,将最小元标识对应的数据添加至该标识对应数据端的最终数据同步列表的末端;更新最小元标识对应数据端的位置指针指向下一位置;在最小堆中删除当前最小元,并增加当前最小元标识对应数据端位置指针所指向的标识,实现最小堆的更新;若更新后最小堆的最小元标识与前一个删除的最小元相等,则重新执行下一轮循环,若不等,则在本轮循环中未添加数据的数据端最终数据同步列表的末端添加随机数据。

[0067] 在具体实施中,如图3所示,所述方法具体包括如下步骤:

[0068] 步骤1:多方数据端A1,A2,…,Ak在各自的本地按标识的大小对数据进行本地排序;

[0069] 其中,所述步骤1的本地排序按照预设的标识进行排序,例如医院的病人信息中,用病人的身份证作为每个人唯一的标识,排序时按照身份证号从小到大进行排序。

[0070] 步骤2:多方数据端在各自本地分别对已排序的数据设置位置指针p1,p2,…,pk,指向各自标识的第一个,即p1=1,p2=1,…,pk=1;

[0071] 步骤3:多方数据端在各自本地初始化最终数据同步列表为空列表,即各自的最终数据同步列表为List1,List2,…,Listk,当前k个列表都为空列表;

[0072] 步骤4:最小堆初始化:多方数据端按照各自位置指针指向的当前标识构造k元最小堆结构;

[0073] 其中,所述步骤4最小堆初始化时,本实施例所述方案采用一个数组来存储这个最小堆,有两种方案可以用来初始化堆,具体的:

[0074] 方案一:不需要维护一个完整的堆结构,而是存储各自端在堆中的位置,即存储自己在堆数组中的位置,以及自己的父结点和子结点。该数据端建堆算法包括如下处理过程:

[0075] (1)交互得到最后一个元素位置,在此位置之后即自己端的位置(不妨记为x),记录下自己的位置;

[0076] (2)记录自己端位置指针指向数据的标识,位置指针向后移动一位,并向父结点使

用基于百万富翁协议的安全计算来比较大小，

[0077] i.若父结点的标识更小，当前端初始化最小堆完成，并向其他所有数据端发送自己的位置；

[0078] ii.若父结点的标识更大，则自己端结点与父结点位置交换，通知所有结点：原父结点对应的数据端更新位置为x。自己端位置更新为父结点位置，并继续步骤(2)；

[0079] (3)自己位置更新完成，通知所有数据端自己的位置；

[0080] (4)下一个数据端继续步骤1，若所有数据端都初始化完成，则最小堆完成，算法结束。

[0081] 需要注意的是，上述方案一不需要每个数据端存储完整的堆，只需要存储自己对应结点的父结点和子结点的数据端编号。如图2所示给出一个用此算法初始化最小堆的过程；开始，只有两个数据端加入了最小堆，一个数据标识为5，一个为6；当第三方加入时，先在最后添加一个新元素，并记录自己的位置和当前指针指向的标识(此例图中显示为2)；向父结点比较大小后得到自己的标识要小，所以结点位置交换，由于没有父结点了，不再向上比较。

[0082] 方案二：所有数据端都各自维护一整个最小堆，由于数据端个数是一个确定的不是很大的常数，所以维护整个最小堆不会有很大的空间复杂度。算法为：每个数据端各自建立一个长度为k(k为数据端个数)的数组，数组中记录的标识统一设置为一个比所有标识都小的标识。例如标识使用的是自然数时，可以设置为0；用身份证号作为标识时，可以设置为-1。

[0083] 步骤5：根据各端中的数据，循环更新最小堆和最终数据同步列表(循环终止条件见步骤7)：

[0084] (1)查看最小堆结构中的最小元，同时最小元标识对应的数据端(不妨设x)在其本地将这一标识相应的数据条目添加到Listx列表的末端；

[0085] (2)更新x的位置指针px，使其指向其本地数据的下一个位置；

[0086] (3)在最小堆结构中，删除最小元，并增加数据端x的标识指针所指向的标识，并更新最小堆；其中，最小堆删除添加元素操作中，由于上述算法删除一个元素后立刻加入一个新的元素，所以此处合并删除添加元素算法为：将最小堆中最小元结点对应的标识更新为新加入的标识，后续再更新这个堆使其成为一个最小堆。

[0087] (4)若更新后最小堆的最小元与前一个删去的最小元相等，继续步骤i；

[0088] (5)若更新后最小堆的最小元与前一个删去的最小元不等，本轮内层循环中未添加数据条目的数据端本地生成随机数据条目添加到各自端列表末端；

[0089] 其中，所述最小堆更新算法：

[0090] (1)若更新的结点有父结点，更新的结点和其父结点对应的数据端交互比较大小；

[0091] i.若更新的结点更大，进入步骤(2)；

[0092] ii.若更新的结点更小，交换此结点与父结点，通知父结点对应的数据端更新自己的位置，回到步骤1继续与父结点比较；

[0093] (2)若更新的结点有子结点，分别与左子结点和右子结点对应的数据端交互比较大小；

[0094] i.若更新的结点更大，交换此结点与该子结点，通知该子结点对应的数据端更新

自己的位置,回到步骤(2)继续与子结点比较;

[0095] ii.若更新的结点更小,进入步骤(3)。

[0096] (3)通知所有数据端自己更新后的位置。

[0097] 最小堆中标识大小的比较使用的为多方安全计算中比较大小的算法,即百万富翁难题的解决方案,下面算法为数据端A向数据端B交互,比较标识keyA和keyB大小:

[0098] (1)数据端A通过百万富翁协议的模块加密自己的标识keyA,记为f(keyA),发送给数据端B;

[0099] (2)数据端B将自己的标识keyB和收到的f(keyA)输入比较大小模块,得出哪一个标识更大,并将标识keyB加密得f(keyB),将f(keyB)发送给数据端A;

[0100] (3)数据端A将自己的标识keyA和收到的f(keyB)输入比较大小模块,得出哪一个标识更大。

[0101] 步骤6:当一个数据端已经遍历到本地列表端结尾,则令其当前标识为无穷大;

[0102] 步骤7:当所有列表的当前标识都为无穷大时,结束循环。

[0103] 在具体实施中,由于数据隐私保护的需求,最小堆结构的实现与传统外排序有所不同,但算法思想一致。最小堆结构的实现:

[0104] 每个数据端需维护当前第一个数据的标识,及其在最小堆中的父结点和子结点所对应的数据端编号;

[0105] 对最小堆进行更新(删除最小元,添加新元)的算法,需通过1中各个数据端存储的父节点和子节点编号,寻找到相应的数据端;

[0106] 不同数据端之间通过编号查找进行两两标识比较;

[0107] 最小堆的两两标识比较算法需要保护数据隐私,因此采用“百万富翁难题”的多方安全计算协议。

[0108] 实施例二:

[0109] 本实施例的目的是提供一种基于堆结构的安全多方数据同步预处理系统。

[0110] 一种基于堆结构的安全多方数据同步预处理系统,其应用于若干数据端之间的多方数据融合,包括:

[0111] 本地排序单元,其用于对于每个数据端,按照其本地数据标识大小对数据进行本地排序,获得本地数据列表;

[0112] 最终数据同步列表初始化单元,其用于对于每个数据端,分别定义指向本地数据列表第一个元素标识的位置指针,并构建最终数据同步列表,其中,所述最终数据同步列表初始为空列表;

[0113] 最小堆构建单元,其用于基于各个数据端的位置指针所指向的标识构建最小堆;

[0114] 多方数据融合单元,其用于通过遍历各个数据端中的本地数据列表,循环更新最小堆以及最终数据同步列表,直至各个数据端中的本地数据列表均遍历至结尾,结束循环;以获得的每个终端的最终数据同步列表作为多方数据的融合结果;

[0115] 其中,所述循环更新最小堆以及最终数据同步列表,具体为:获取最小堆结构中的最小元,将最小元标识对应的数据添加至该标识对应数据端的最终数据同步列表的末端;更新最小元标识对应数据端的位置指针指向下一位置;在最小堆中删除当前最小元,并增加当前最小元标识对应数据端位置指针所指向的标识,实现最小堆的更新;若更新后最小

堆的最小元标识与前一个删除的最小元相等,则重新执行下一轮循环,若不等,则在本轮循环中未添加数据的数据端最终数据同步列表的末端添加随机数据。

[0116] 在更多实施例中,还提供:

[0117] 一种电子设备,包括存储器和处理器以及存储在存储器上并在处理器上运行的计算机指令,所述计算机指令被处理器运行时,完成实施例一中所述的方法。为了简洁,在此不再赘述。

[0118] 应理解,本实施例中,处理器可以是中央处理单元CPU,处理器还可以是其他通用处理器、数字信号处理器DSP、专用集成电路ASIC,现成可编程门阵列FPGA或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件等。通用处理器可以是微处理器或者该处理器也可以是任何常规的处理器等。

[0119] 存储器可以包括只读存储器和随机存取存储器,并向处理器提供指令和数据、存储器的一部分还可以包括非易失性随机存储器。例如,存储器还可以存储设备类型的信息。

[0120] 一种计算机可读存储介质,用于存储计算机指令,所述计算机指令被处理器执行时,完成实施例一中所述的方法。

[0121] 实施例一中的方法可以直接体现为硬件处理器执行完成,或者用处理器中的硬件及软件模块组合执行完成。软件模块可以位于随机存储器、闪存、只读存储器、可编程只读存储器或者电可擦写可编程存储器、寄存器等本领域成熟的存储介质中。该存储介质位于存储器,处理器读取存储器中的信息,结合其硬件完成上述方法的步骤。为避免重复,这里不再详细描述。

[0122] 本领域普通技术人员可以意识到,结合本实施例描述的各示例的单元即算法步骤,能够以电子硬件或者计算机软件和电子硬件的结合来实现。这些功能究竟以硬件还是软件方式来执行,取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能,但是这种实现不应认为超出本公开的范围。

[0123] 上述实施例提供的一种基于堆结构的安全多方数据同步预处理方法及系统可以实现,具有广阔的应用前景。

[0124] 以上所述仅为本公开的优选实施例而已,并不用于限制本公开,对于本领域的技术人员来说,本公开可以有各种更改和变化。凡在本公开的精神和原则之内,所作的任何修改、等同替换、改进等,均应包含在本公开的保护范围之内。

A

key_1	key_2	key_6	key_7	key_9
dataA1	dataA2	dataA6	dataA7	dataA9

B

key_2	key_3	key_5	key_8	key_9
dataB2	dataB3	dataB5	dataB8	dataB9

C

key_2	key_4	key_7	key_8	key_9
dataC2	dataC4	dataC7	dataC8	dataC9

多方数据融合

A

key_1	key_2	*	*	*	key_6	key_7	*	key_9
dataA1	dataA2	*	*	*	dataA6	dataA7	*	dataA9

B

*	key_2	key_3	*	key_5	*	*	key_8	key_9
*	dataB2	dataB3	*	dataB5	*	*	dataB8	dataB9

C

*	key_2	*	key_4	*	*	key_7	key_8	key_9
*	dataC2	*	dataC4	*	*	dataC7	dataC8	dataC9

图1

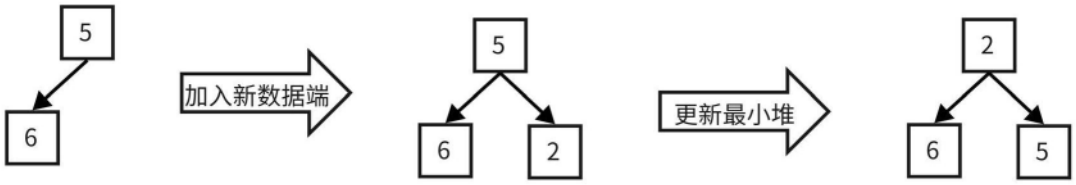


图2

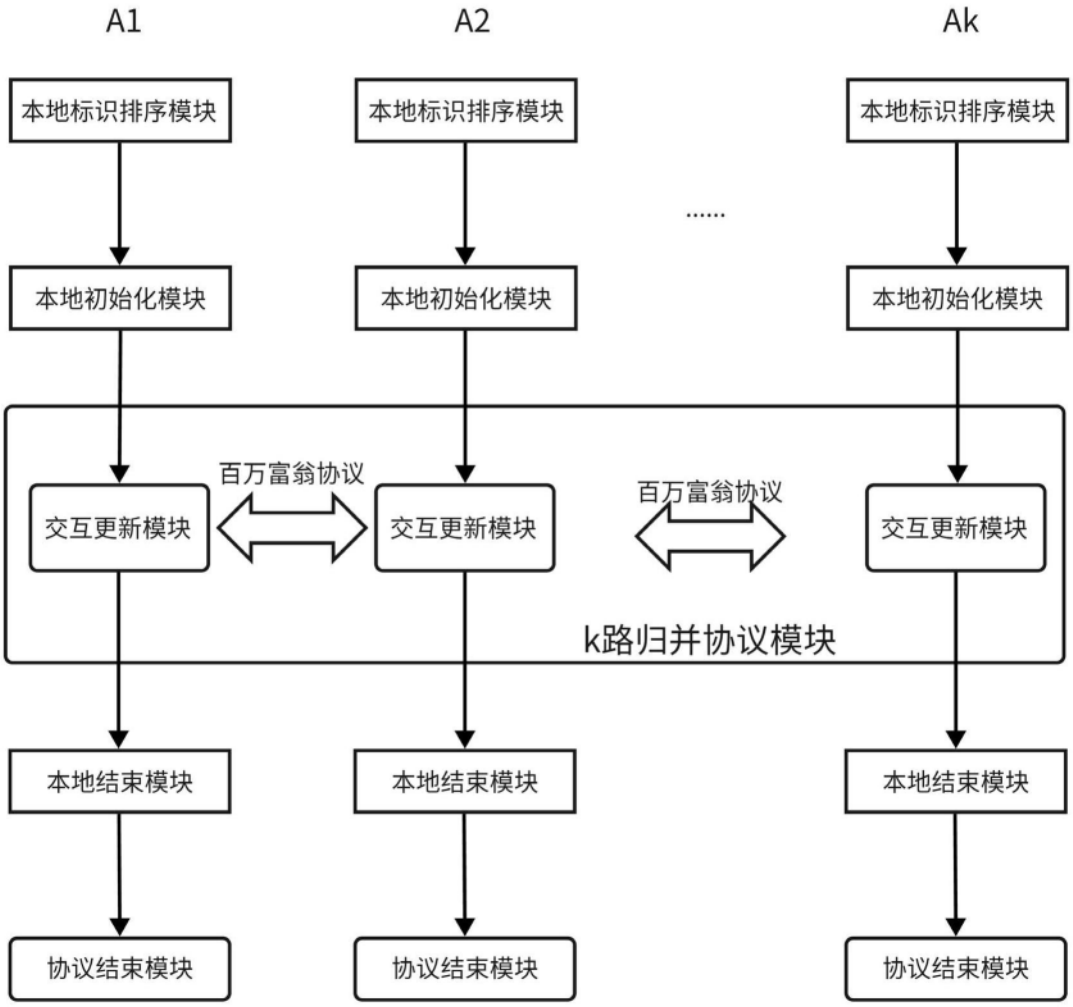

A1
A2
Ak
本地标识排序模块
本地标识排序模块
本地标识排序模块
.....
本地初始化模块
本地初始化模块
本地初始化模块
百万富翁协议
百万富翁协议
交互更新模块
交互更新模块
交互更新模块
k路归并协议模块
本地结束模块
本地结束模块
本地结束模块
协议结束模块
协议结束模块
协议结束模块

图3